Polynomial Algorithms for the Synthesis of Hazard-free Circuits from Signal Transition Graphs *

Enric Pastor and Jordi Cortadella
Department of Computer Architecture
Universitat Politècnica de Catalunya
08071 Barcelona, Spain

Abstract

Methods for the synthesis of asynchronous circuits from Signal Transition Graphs (STGs) have commonly used the State Graph to solve the two main steps of this process: the state assigment problem and the generation of hazard-free logic. The size of the State Graph can be of order $O(2^n)$, where n is the number of signals of the circuit. As synthesis tools for asynchronous systems start to mature, the size of the STGs increases and the exponential algorithms that work on the State Graph become obsolete. This paper presents alternative algorithms that work in polynomial time and, therefore, avoid the generation of the SG. With the proposed algorithms, STGs can be synthesized and hazard-free circuits generated in extremely low CPU times. Improvements in 2 or 3 orders of magnitude (from hours to seconds) with respect to existing algorithms are achieved when synthesizing fairly large STGs.

1 Introduction

Asynchronous circuits have gained interest in the last few years, specially in the area of interface circuits. However they have not been widely used due to the difficulty of design and the lack of synthesis and verification tools. Signal Transition Graphs (STGs) have been proposed as a specification formalism for asynchronous control circuits [1, 13].

Most current designs based on STG specifications have been handcrafted and, therefore, their complexity is relatively manageable by designers. Even for some of such descriptions, the synthesis algorithms take a significant CPU time, of the order of several minutes or hours. This results directly from the fact that the number of states derived from an STG rapidly exploits with the number of signals and the degree of parallelism intrinsic to the underlying net. Similarly to what happened with synchronous FSMs, as high-level synthesis tools start becoming mature, asynchronous FSMs will be automatically generated to specify the behavior of interface and data-path controllers [3] and its complexity will be no longer manageable by exponential algorithms.

Therefore, there is a need to devise algorithms whose complexity depends on the size of the description rather than in the number of states. This paper aims at making some key contributions in this direction. The algorithms proposed to verify the Complete State Coding (CSC) problem and to synthesize hazard-free circuits must, first, avoid the enumeration of the states generated by the STG description and then, avoid net traversals that depend on the number of states of the STG. In this way, polynomial algorithms are proposed.

The two-level circuits generated by the proposed polynomial algorithms can be valid implementations of the specifications if the environment behaves as assumed by the STG. Furthermore, this implementation can be transformed into a multi-level logic circuit with more general delay models by using existing techniques [6].

1.1 Previous work

Methods for solving the CSC problem in STGs have been proposed in [7, 14, 15, 16]. Vanbekbergen [14] and Ykman [16] have presented algorithms that work directly on the STG, but only in [14] a polynomial-time technique based on the lock graph theory has been proposed. However, this technique is only applicable to marked graphs with single transitions. The other approaches require the construction of the State Graph (SG), which can have a number of states of $O(2^n)$, being n the number of signals.

Algorithms for hazard-free synthesis from STGs have been proposed by Chu [2], Moon et al. [9], and Lavagno et al. [5] assuming different delay models and modes of operation. The most general delay model (bounded wire delay) has been considered in [6] where hazards are eliminated by solving a linear programming problem.

However, all the proposed approaches require an exhaustive analysis of the states of the system either at the level of State Graph (SG) or next-state tables and, therefore, their corresponding algorithms are exponential.

2 Definitions

This section presents some basic definitions and notations used along the paper. We have imported other preliminary definitions and results from [1, 4, 7, 10, 14].

Signal Transition Graphs (STGs) are Petri nets, whose transitions are interpreted as value changes on signals of the circuit. Rising and falling transitions for signal t are denoted by t^+ and t^- respectively, while t^* denotes an up- or down-transition. S_a denotes the set of all signals in the STG, and S_{NI} denotes the set of non-input signals.

*Work funded by CYCIT TIC 91-1036, ACiD-WG (Esprit 7225) and Dept. d'Ensenyament de la Generalitat de Catalunya

If the STG is live, each state (marking) s_i of the State Graph is assigned a binary vector of signal values (state code). The binary vector $v_i \in \{0,1\}^n$ $(n = |S_a|)$ is the code of state s_i, while v_i^j denotes the value of signal t_j in the state s_i. If $M = \{m_1, \ldots, m_n\}$ is a set of markings of an STG, then $\widehat{M} = \{v_1, \ldots, v_n\}$ denotes the set of state codes of the SG corresponding to the markings in M.

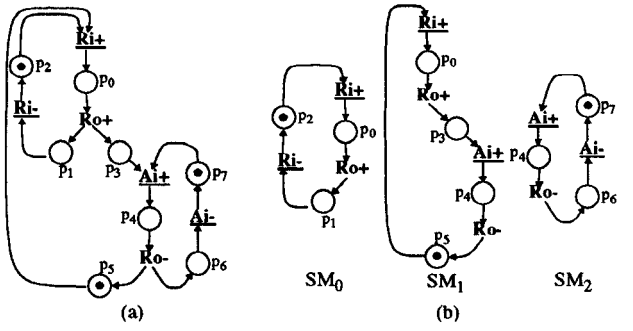

Figure 1: (a) STG Specification of a PLA interface circuit, (b) SMC descomposition

Figure 1(a) depicts an example of STG with two input signals (*Ri* and *Ai*) and one output signal (*Ro*), and its initial marking. This example has been taken from [7] and specifies the behavior of a PLA interface circuit. Figure 1(b) shows the State Machine Components (SMCs) that cover all the transitions for each signal in the STG.

3 A P-time Algorithm for CSC Verification

Next we present a novel approach to verify the CSC property for Free-choice STGs. Previously, we propose an exponential method that operates directly on the STG. Then we make it polynomial by using some relaxations.

Definition 1 *Given an STG S and a non-input signal t_i, we define*

$$\mathcal{M}_i^+ = \{m_j \in [m_0\rangle \mid \exists t_i^+ : m_j[t_i^+\rangle\}$$
$$\mathcal{M}_i^1 = \{m_j \in [m_0\rangle \mid v_j^i = 1 \wedge \nexists t_i^- : m_j[t_i^-\rangle\}$$
$$\mathcal{M}_i^- = \{m_j \in [m_0\rangle \mid \exists t_i^- : m_j[t_i^-\rangle\}$$
$$\mathcal{M}_i^0 = \{m_j \in [m_0\rangle \mid v_j^i = 0 \wedge \nexists t_i^+ : m_j[t_i^+\rangle\}$$

$\mathcal{M}_i^+$ ($\mathcal{M}_i^-$) is the set of markings in which some t_i^+ (t_i^-) is enabled. $\mathcal{M}_i^1$ ($\mathcal{M}_i^0$) is the set of markings in which the corresponding state code is such that $v_j^i = 1$ ($v_j^i = 0$) and no t_i^* transition is enabled.

Theorem 2 [12] *An STG S has the CSC property iff*

$$\forall t_i \in S_{NI} \quad \widehat{\mathcal{M}_i^+} \cap \widehat{\mathcal{M}_i^0} = \emptyset \ \wedge \ \widehat{\mathcal{M}_i^-} \cap \widehat{\mathcal{M}_i^1} = \emptyset$$

The calculation of the $\mathcal{M}_i$ sets requires a complete reachability analysis of the net. The approach we propose avoids the exponentiality of this analysis by overestimating the $\widehat{\mathcal{M}_i}$ sets.

Definition 3 *For each place p and transition t of an STG, we define*

$$\mathcal{V}_p = \{m_j \in [m_o\rangle \mid m_j(p) = 1\}$$
$$\mathcal{V}_t = \{m_j \in [m_o\rangle \mid m_j[t\rangle\}$$

Definition 4 *Given a place p and a transition t, we define $\mathcal{C}_p$ and $\mathcal{C}_t$ as the smallest cubes (i.e. with the greatest number of literals) that cover $\widehat{\mathcal{V}_p}$ and $\widehat{\mathcal{V}_t}$ respectively.*

$\mathcal{C}_p$ and $\mathcal{C}_t$ can be calculated in polynomial time for free-choice nets by using the algorithm presented in [11].

Definition 5 *A set of SMCs, $\Sigma = \{S_i\}$, is an* SM-cover *of an STG S if all members of Σ are SMCs of S and all the places and transitions of S are covered by some S_i. An SM-cover Σ is* irredundant *if no subset of Σ is an SM-cover.*

Finding an SMC that covers a place or a transition can be done in polynomial time by using the extension of Hack's algorithm [4] presented in [11]. Finding an *irredundant cover* can be done by iterative generation of SMCs that cover places or transitions not covered by previous SMCs.

The following algorithm makes a conservative verification of the CSC property, i.e. if a CSC conflict exists then it is detected by the algorithm. On the other hand, the use of the $\mathcal{C}$ cubes instead of the $\mathcal{M}$ sets may produce the detection of non-existing conflicts as "potential conflicts". State signals are inserted to disambiguate the CSC conflicts using the approach presented in [11].

```
has_csc_property (S) {
    Let Σ = {S_m} be an irredundant SM-cover of S
    foreach S_m ∈ Σ do
      foreach pair p_j and p_k covered by S_m s.t. j > k do
        foreach non-input signal t_i do
          if (∃t_i* ∈ p_j•) ∧ (C_{p_k} ∩ C_{t_i*} ≠ ∅) then return false;
          if (∃t_i* ∈ p_k•) ∧ (C_{p_j} ∩ C_{t_i*} ≠ ∅) then return false;
    return true;
}
```

4 Hazard-free Next-state Function

Once an STG satisfies the CSC property, a hazard-free logic circuit can be derived. In this section a polynomial time algorithm for the synthesis of hazard-free two-level circuits under Multiple Signal Change (MSC) conditions and *unbounded gate-delay* model (speed independent) is presented. Additionally, a *well-behaved* environment is assumed [8] (it does not apply inputs too quickly so that the circuit can become stable). This circuit can be a starting point for other synthesis steps oriented to multi-level logic and/or more general delay models [6].

The proposed approach assumes that the CSC property has been guaranteed by using the CSC solving procedure presented in the previous section. Moreover, it requires the SM-cover used for CSC verification to be *well-formed* for synthesis.

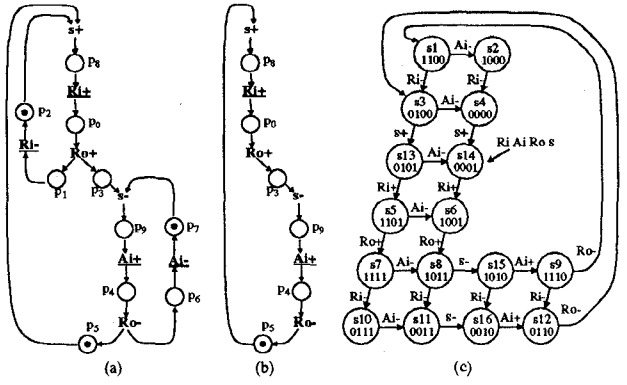

Figure 2: (a)PLA interface circuit with the CSC property, (b) SMC for signals *Ro* and *s*, (c) State Graph

Definition 6 *Given an STG S, an SM-cover* $\Sigma = \{S_k\}$ *is said to be* well-formed *for synthesis if for any* $t_i \in S_{NI}$ *there is an* $S_i \in \Sigma$ *such that all transitions of signal* t_i *are covered by* S_i.

From now on, we will denote by S_i an SMC that covers all transitions of signal t_i (note that one SMC can cover more than one signal). Tables 1 and 2 show the $\mathcal{V}$ sets and their corresponding cubes for the encoded PLA interface circuit depicted in Figure 2(a). The SMC that covers all transitions of signals *Ro* and *s* is in Figure 2(b) and the State Graph is in Figure 2(c).

p	$\mathcal{V}_p$	$\mathcal{C}_p$ (Ri Ai Ro s)
p_0	{m5, m6}	1 - 0 1
p_1	{m1, m2, m7, m8, m9, m15}	1 - - -
p_2	{m3, m4, m10, m11, m12, m16}	0 - - -
p_3	{m7, m8, m10, m11}	- - 1 1
p_4	{m9, m12}	- 1 1 0
p_5	{m1, m2, m3, m4}	- - 0 0
p_6	{m1, m3, m5, m7, m10, m13}	- 1 - -
p_7	{m2, m4, m6, m8, m11, m14}	- 0 - -
p_{c1}	{m13, m14}	0 - 0 1
p_{c2}	{m15, m16}	- 0 1 0

Table 1: $\mathcal{V}_p$ and $\mathcal{C}_p$ for the places of the example

t	$\mathcal{V}_t$	$\mathcal{C}_t$ (Ri Ai Ro s)
R_i+	{m13, m14}	0 - 0 1
R_o+	{m5, m6}	1 - 0 1
R_i-	{m1, m2, m7, m8, m9, m15}	1 - - -
A_i+	{m15, m16}	- 0 1 0
R_o-	{m9, m12}	- 1 1 0
A_i-	{m1, m3, m5, m7, m10, m13}	- 1 - -
s+	{m3, m4}	0 - 0 0
s-	{m8, m11}	- 0 1 1

Table 2: $\mathcal{V}_t$ and $\mathcal{C}_t$ for the transitions of the example

4.1 Deriving Logic from a State Machine

We will denote by F_i and R_i the *on-set* and *off-set* covers derived for the next-state function of the non-input signal t_i. The on-set and the off-set of the next-state function f_i can be expressed in terms of the $\widehat{\mathcal{M}}_i$ sets:

$$\text{On-set}(f_i) = \widehat{\mathcal{M}}_i^+ \cup \widehat{\mathcal{M}}_i^1$$

$$\text{Off-set}(f_i) = \widehat{\mathcal{M}}_i^- \cup \widehat{\mathcal{M}}_i^0$$

Instead, we calculate covers by using the cubes $\mathcal{C}_p$ and $\mathcal{C}_t$ for all places and transitions of signal t_i of S_i.

Given an SMC S_i that covers signal t_i, the places covered by S_i can be partitioned into four sets: P_i^1 (places between a t_i^+ and a t_i^- transitions and not predecessor of any t_i^- transition), P_i^0 (places between a t_i^- and a t_i^+ transitions and not predecessor of any t_i^+ transition), P_i^+ (places predecessor of a t_i^+ transition) and, P_i^- (places predecessor of a t_i^- transition).

Clearly, the state codes of the markings that have a token in some place in P_i^1 belong to the on-set of f_i. These codes belong to the corresponding $\widehat{\mathcal{V}}_p$ sets and are covered by the $\mathcal{C}_p$ cubes.

The state codes in $\widehat{\mathcal{V}}_{t_i+}$ for any rising transition of t_i belong to the on-set of f_i. These codes are covered by the $\mathcal{C}_{t_i+}$ cubes.

Finally, the state codes in $\widehat{\mathcal{V}}_p - \widehat{\mathcal{V}}_{t_i-}$ for any $p \in P_i^-$ and predecessor of t_i-, belong to the on-set of f_i. These codes are *also covered* by $D = \mathcal{C}_p - \mathcal{C}_{t_i-}$. Note that D is not a cube, but a cover. Its calculation can be done in polynomial time and, the fact that D covers $\widehat{\mathcal{V}}_p - widehat\mathcal{V}_{t_i-}$ results from the CSC verification procedure of the previous section as shown in [12].

Thus, an initial F_i cover can be obtained by using this approach (similarly for an initial R_i cover). For hazard elimination, consensus between cubes corresponding to pairs of adjacent places (separated by one transition) and pairs of adjacent places and transitions are properly added. Finally, a prime cover is obtained by expanding the resulting cubes in F_i against R_i.

The following algorithm obtains a SOP hazard-free cover F_i for the next-state function f_i.

```
hazard-free_SOP_implementation {
    let S_i be an SMC that covers signal t_i
    F_i = R_i = ∅

    /* Calculate an initial cover for F_i */
    foreach place p_j covered by S_i do
        if (∃t = t_i^- ∈ p_j^•) D_{p_j} = C_{p_j} - C_t
        else if (∃t = t_i^+ ∈ p_j^•) D_{p_j} = C_t
            else D_{p_j} = C_{p_j}
        F_i = F_i ∪ D_{p_j}

    /* Calculate an initial cover for R_i */
    ... Similarly to F_i ...

    /* Add consensus for hazard elimination */
    foreach transition t covered by S_i do
        let p_k be the place in the inset of t covered by S_i
        let p_l be the place in the outset of t covered by S_i
        F_i = F_i ∪ Consensus(D_{p_k}, D_{p_l})
        /* if some D contains more than one cube,
           consensus is generated for all cubes */

    F_i = expand(F_i, R_i)
    /* F_i is expanded against R_i,
       and cubes included in other cubes are eliminated */
}
```

F_{Ro}	$C_{Ro+} \cup C_{p_3} \cup C_{p_{c2}} \cup (C_{p_4} - C_{Ro-}) \cup$ $(C_{Ro+} \odot C_{p_3}) \cup (C_{p_3} \odot C_{p_{c2}}) \cup (C_{p_{c2}} \odot (C_{p_4} - C_{Ro-}))$	$Ri\ Ro's + Ro\ s + Ai'Ro\ s'$ $+Ri\ s + Ai'Ro$
R_{Ro}	$C_{Ro-} \cup C_{p_5} \cup C_{p_{c1}} \cup (C_{p_0} - C_{Ro+}) \cup$	$Ai\ Ro\ s' + Ro's' + Ri'Ro's$
F_s	$C_{s+} \cup C_{p_{c1}} \cup C_{p_0} \cup (C_{p_3} - C_{s-})$ $(C_{s+} \odot C_{p_{c1}}) \cup (C_{p_{c1}} \odot C_{p_0}) \cup (C_{p_0} \odot (C_{p_3} - C_{s-}))$	$Ri'Ro's' + Ri'Ro's + Ri\ R'_o s + Ai'Ro's'$ $+Ri'Ro' + Ro's + Ri\ Ai\ s$
R_s	$C_{s-} \cup C_{p_{c2}} \cup C_{p_4} \cup (C_{p_5} - C_{s+})$	$Ai'Ro\ s + Ai'Ro\ s' + Ai\ Ro\ s' + Ri\ Ro's'$

Table 3: F and R covers for signals R_o and s ($\odot$ stands for consensus)

Table 3 presents the F and R covers calculated by the synthesis procedure before expanding F against R. The final result of the synthesis is:

$$F_{R_o} = R_o s + R_i s + A'_i$$
$$F_s = R'_o s + A_i s + R'_i R'_o$$

Theorem 7 [12] *The F_i and R_i covers obtained by the synthesis procedure are valid on-set and off-set covers of f_i, i.e. every on-set (off-set) vertex and no off-set (on-set) vertex of f_i is covered by a cube in F_i (R_i).*

4.2 Hazard-freeness under MSC Condition

Theorem 8 [12] *The two-level sum-of-product implementation obtained by the synthesis procedure is free of hazards under MSC conditions.*

Proof: Let us assume that s_1 is the state in which a set T of concurrent transitions is enabled. Let s_2 be the state reached from s_1 after firing all transitions in T. Let us define C_T, the transition cube for T, as follows:

$$C_T^j = \begin{cases} - & \text{if some } t_j^* \in T \\ v_1^j & \text{otherwise} \end{cases}$$

Since T is a set of concurrent transitions (they can be fired in any order), all states covered by C_T are valid states of the SG. Let us assume we are deriving an *on-set* cover for signal t_i by using the SMC S_i according to synthesis procedure.

$0 \Rightarrow 0$ transitions. All states covered by C_T belong to the off-set of f_i. Therefore, no cube of F_i covers any of them, otherwise the STG would not have the CSC property. Hence, no 1-hazards can be produced.

$0 \Rightarrow 1$ transitions. All states covered by C_T belong to the off-set of f_i, excepting s_2. Therefore, there is at least one cube of F_i that covers s_2 and no cube of F_i that covers the rest of states covered by C_T, otherwise the STG would not have the CSC property. Hence, no hazards can be produced.

$1 \Rightarrow 1$ transitions. Let us call m_1 and m_2 the markings corresponding to states s_1 and s_2. There is a place p_1 covered by S_i such that $m_1(p_1) = 1$. Let us call t the transition in the outset of p_1 covered by S_i. If $t \notin T$ then C_T is completely covered by C_{p_1} and no 0-hazards can be produced. If $t \in T$, let us call p_2 the place in the outset of t covered by S_i. Hence $m_2(p_2) = 1$. According to the synthesis procedure, C_T will be covered by $C_{p_1} \cup C_{p_2}$. Since C_T is a cube, then C_T is completely covered by the $consensus(C_{p_1}, C_{p_2})$ and thus no 0-hazards can be produced.

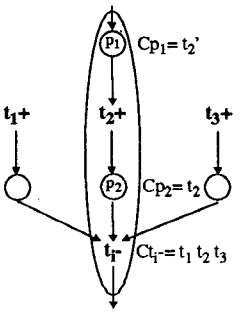

Figure 3: Hazard-free $1 \rightarrow 0$ transition for signal t_i

$1 \Rightarrow 0$ transitions. We will prove this case, without loss of generality, by means of the example shown in Figure 3. State s_2 (with code v_2) is the one in which t_i- is enabled. t_1+, t_2+, and t_3+ are the set of concurrent transitions T. The synthesis procedure will include the following cubes to F_i: C_{p_1}, $C_{p_2} - C_{t_i-}$, and $consensus(C_{p_1}, C_{p_2} - C_{t_i-})$. If we only consider those literals involved in the transitions of the example, the cubes mentioned before have the following literals: $C_{p_1} = t'_2$, $C_{p_2} - C_{t_i-} = t'_1 t_2 + t'_3 t_2$, and $consensus(C_{p_1}, C_{p_2} - C_{t_i-}) = t'_1 + t'_3$. After eliminating cubes included in other cubes, the resulting cubes will be t'_1, t'_2, and t'_3. This three cubes cover $C_T - v_2$ and, furthermore, they change monotonically from 1 to 0 as the corresponding transition fires. Hence, no hazards can be produced. This proof can be easily extended to any number of concurrent transitions. □

5 Experimental results

The examples used for our experiments have been obtained from [7]. Moreover, we have also included some STGs automatically generated by high-level synthesis tools (gcd_alu, gcd_reg_a).

Table 4 presents de results obtained by *sis* [7] and our approach. The number of states of the original STG, the number of signals and transitions before and after state assigment and the number of literals of the resulting cover are presented. CPU time is given in seconds. Results for *sis* synthesizing gcd_alu and gcd_reg_a have been obtained in our machine (SPARC IPX with 32 Mbyte main memory). The other results for *sis* have been obtained from [7].

	initial			final [7]				final (this paper)			
STG	sig	tr	states	sig	tr	lit	CPU time	sig	tr	lit	CPU time
alloc-outbound	7	18	17	9	22	19	6	9	22	28	1.5
atod	6	12	20	7	14	14	5.2	7	14	20	0.7
nak-pa	9	18	56	10	22	30	9.9	10	20	32	1.9
ram-read-sbuf	10	20	36	11	22	20	8	11	22	40	2.7
sbuf-ram-write	10	20	58	12	24	30	11.5	12	24	43	3.6
sbuf-read-ctl	6	12	14	7	14	13	5.2	7	14	11	1.1
sendr-done	3	6	7	4	8	5	3.5	4	8	6	0.2
vbe4a	6	12	76	8	16	22	10.1	8	16	25	0.8
vbe6a	8	16	128	10	20	30	18.4	10	20	48	1.4
master-read	13	26	8932	16	36	77	1635.1	15	30	46	3.5
gcd_alu	8	16	228	12	24	41	344.0	12	24	84	3.2
gcd_reg_a	20	58	1274	25	75	84	5648.0	25	75	124	72.2

Table 4: Experimental Results

Two conclusions can be derived from the results:

- The CPU times obtained by *sis* grow exponentially with the size of the STG. Exponential algorithms will become obsolete for large STGs.
- The size of the circuit obtained by *sis* is usually smaller due to the relaxations introduced in our algorithms to maintain their polynomial complexity.

The most significant exception is the master_read example, where we obtain a two-level cover of 46 literals and *sis* generates 77 literals. Furthermore, only 2 state signals are inserted with our approach.

The key result is that for moderately large STGs (master_read, gcd_alu, and gcd_reg_a) the amount of CPU time spent by the polynomial algorithm is of 2-3 orders of magnitude smaller than that used by exponential algorithms.

6 Conclusions

Polynomial algorithms for the synthesis of asynchronous circuits from STGs have been presented in this paper.

It seems that polynomial algorithms are the only valid approach to manage fairly large examples with moderate CPU times. The quality of the obtained circuits is comparable to those obtained by exponential techniques. However they are usually larger due to the relaxations introduced in the algorithms to make them polynomial.

Current efforts are directed towards extending the proposed techniques to more general delay models (bounded and unbounded wire delays) and to wider classes of Petri nets. Also, more accurate heuristics to improve the quality of the circuits are explored.

References

[1] Tam-Anh Chu. *Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications.* Ph.d. thesis, MIT, June 1987.

[2] Tam-Anh Chu. Automatic synthesis and verification of hazard-free control circuits from asynchronous finite state machine specifications. In *Proc. of the ICCD*, pages 407–413, October 1992.

[3] Jordi Cortadella and Rosa M. Badia. An asynchronous architecture model for behavioral synthesis. In *Proc. of EDAC*, pages 307–311, March 1992.

[4] M. Hack. *Analysis of production schemata by Petri nets.* M.s. thesis, MIT, February 1972.

[5] L. Lavagno, K. Keutzer, and A. Sangiovanni-Vincentelli. Algorithms for synthesis of hazard-free asynchronous circuits. In *Proc. of the 28th. DAC*, pages 302–308, June 1991.

[6] L. Lavagno and A. Sangiovanni-Vincentelli. Linear programming for optimum hazard elimination in asynchronous circuits. In *Proc. of the ICCD*, pages 275–278, October 1992.

[7] Luciano Lavagno, Cho W. Moon, Robert K. Brayton, and A. Sangiovanni-Vincentelli. Solving the state assignment problem for signal transition graphs. In *Proc. of the 29th. DAC*, pages 568–572, June 1992.

[8] Cho W. Moon. *Synthesis and Verification of Asynchronous Circuits from Graphical Specifications.* Ph.d. thesis, UCB/ERL, 1992.

[9] Cho W. Moon, Paul R. Stephan, and Robert K. Brayton. Synthesis of hazard-free asynchronous circuits from graphical specifications. In *Proc. of the ICCAD*, pages 322–325, November 1991.

[10] Tadao Murata. Petri Nets: Properties, analysis and applications. *Proceedings of the IEEE*, Vol. 77(4):541–574, April 1989.

[11] Enric Pastor and Jordi Cortadella. An efficient unique state coding algorithm for signal transition graphs. In *Proc. of the ICCD*, October 1993.

[12] Enric Pastor and Jordi Cortadella. Polynomial algorithms for complete state coding and synthesis of hazard-free circuits from signal transition graphs. Technical Report RR-93/17, UPC/DAC, September 1993.

[13] L. Ya. Rosenblum and A. V. Yakovlev. Signal graphs: From self-timed to timed ones. In *International Workshop on Timed Petri Nets*, pages 199–206, 1985.

[14] P. Vanbekbergen. Optimized synthesis of asynchronous control circuits from graph-theoretic specification. In *Proc. of the ICCAD*, pages 184–187, November 1990.

[15] P. Vanbekbergen, B. Lin, G. Goossens, and H. De Man. A generalized state assignment theory for transformations on signal transition graphs. In *Proc. of the ICCAD*, pages 112–117, November 1992.

[16] Chantal Ykman-Couvreur, Bill Lin, Gert Goossens, and Hugo De Man. Synthesis and optimization of asynchronous controllers based on extended lock graph theory. In *EDAC*, pages 512–517, February 1993.